import React, { useState, useRef, useEffect } from 'react';

import {

MapPin, Circle, Square, Pencil, Type, Eraser,

RotateCcw, Save, Download, Printer, Layers,

ZoomIn, ZoomOut, Eye, ChevronLeft, ChevronRight,

Activity, Target, AlertTriangle, Droplet,

Maximize2, Move, Trash2, Copy, Info

} from 'lucide-react';

const BodyMappingInterface = () => {

const canvasRef = useRef(null);

const [currentView, setCurrentView] = useState('anterior');

const [currentTool, setCurrentTool] = useState('marker');

const [markerType, setMarkerType] = useState('pain');

const [markers, setMarkers] = useState([]);

const [showLayers, setShowLayers] = useState(false);

const [layers, setLayers] = useState({

surface: true,

muscular: false,

skeletal: false,

nervous: false

});

const [selectedMarker, setSelectedMarker] = useState(null);

const [zoom, setZoom] = useState(1);

const [isDrawing, setIsDrawing] = useState(false);

const [history, setHistory] = useState([]);

// Tipi di marcatori disponibili

const markerTypes = {

pain: { icon: '✕', color: '#FF0000', name: 'Dolore' },

trigger: { icon: '●', color: '#FF6B6B', name: 'Trigger Point' },

treatment: { icon: '○', color: '#4ECDC4', name: 'Area Trattamento' },

inflammation: { icon: '▲', color: '#FFD93D', name: 'Infiammazione' },

scar: { icon: '—', color: '#6C5CE7', name: 'Cicatrice' },

radiating: { icon: '↗', color: '#FFA500', name: 'Dolore Irradiato' },

numbness: { icon: '░', color: '#95A5A6', name: 'Parestesia' },

swelling: { icon: '◎', color: '#3498DB', name: 'Edema' }

};

// Template preimpostati

const templates = {

lombalgia: {

name: 'Lombalgia',

markers: [

{ x: 200, y: 280, type: 'pain', intensity: 7 },

{ x: 180, y: 290, type: 'trigger', intensity: 5 },

{ x: 220, y: 290, type: 'trigger', intensity: 5 }

]

},

cervicalgia: {

name: 'Cervicalgia',

markers: [

{ x: 200, y: 120, type: 'pain', intensity: 6 },

{ x: 170, y: 130, type: 'trigger', intensity: 4 },

{ x: 230, y: 130, type: 'trigger', intensity: 4 }

]

},

spalla: {

name: 'Spalla',

markers: [

{ x: 140, y: 150, type: 'pain', intensity: 8 },

{ x: 130, y: 160, type: 'inflammation', intensity: 6 }

]

}

};

// Gestione click su canvas

const handleCanvasClick = (e) => {

if (currentTool !== 'marker') return;

const rect = canvasRef.current.getBoundingClientRect();

const x = e.clientX - rect.left;

const y = e.clientY - rect.top;

const newMarker = {

id: Date.now(),

x: x,

y: y,

type: markerType,

view: currentView,

intensity: 5,

notes: '',

timestamp: new Date().toISOString()

};

setMarkers([...markers, newMarker]);

saveToHistory();

};

// Salva stato per undo

const saveToHistory = () => {

setHistory([...history, markers]);

};

// Undo ultima azione

const handleUndo = () => {

if (history.length > 0) {

const newHistory = [...history];

const previousState = newHistory.pop();

setMarkers(previousState);

setHistory(newHistory);

}

};

// Applica template

const applyTemplate = (templateName) => {

const template = templates[templateName];

if (template) {

saveToHistory();

setMarkers([...markers, ...template.markers.map(m => ({

...m,

id: Date.now() + Math.random(),

view: currentView,

notes: '',

timestamp: new Date().toISOString()

}))]);

}

};

// Componente Body SVG

const BodySVG = ({ view }) => {

const getBodyPath = () => {

if (view === 'anterior') {

return (

<g>

{/\* Testa \*/}

<ellipse cx="200" cy="50" rx="30" ry="35" fill="#FDB5A6" stroke="#333" strokeWidth="2"/>

{/\* Collo \*/}

<rect x="185" y="80" width="30" height="25" fill="#FDB5A6" stroke="#333" strokeWidth="1"/>

{/\* Torso \*/}

<rect x="160" y="105" width="80" height="120" rx="5" fill="#FDB5A6" stroke="#333" strokeWidth="2"/>

{/\* Braccia \*/}

<rect x="110" y="110" width="25" height="100" rx="5" fill="#FDB5A6" stroke="#333" strokeWidth="2" transform="rotate(-5 122 160)"/>

<rect x="265" y="110" width="25" height="100" rx="5" fill="#FDB5A6" stroke="#333" strokeWidth="2" transform="rotate(5 277 160)"/>

{/\* Mani \*/}

<ellipse cx="105" cy="215" rx="15" ry="20" fill="#FDB5A6" stroke="#333" strokeWidth="1"/>

<ellipse cx="295" cy="215" rx="15" ry="20" fill="#FDB5A6" stroke="#333" strokeWidth="1"/>

{/\* Bacino \*/}

<rect x="165" y="225" width="70" height="40" rx="5" fill="#FDB5A6" stroke="#333" strokeWidth="2"/>

{/\* Gambe \*/}

<rect x="170" y="265" width="28" height="110" rx="5" fill="#FDB5A6" stroke="#333" strokeWidth="2"/>

<rect x="202" y="265" width="28" height="110" rx="5" fill="#FDB5A6" stroke="#333" strokeWidth="2"/>

{/\* Piedi \*/}

<ellipse cx="184" cy="385" rx="18" ry="12" fill="#FDB5A6" stroke="#333" strokeWidth="1"/>

<ellipse cx="216" cy="385" rx="18" ry="12" fill="#FDB5A6" stroke="#333" strokeWidth="1"/>

{/\* Linee anatomiche di riferimento \*/}

{layers.skeletal && (

<g opacity="0.3">

<line x1="200" y1="105" x2="200" y2="265" stroke="#666" strokeDasharray="2,2"/>

<circle cx="200" cy="150" r="3" fill="#666"/>

<circle cx="200" cy="200" r="3" fill="#666"/>

<circle cx="200" cy="250" r="3" fill="#666"/>

</g>

)}

</g>

);

} else if (view === 'posterior') {

return (

<g>

{/\* Vista posteriore - simile ma con dettagli diversi \*/}

<ellipse cx="200" cy="50" rx="30" ry="35" fill="#FDB5A6" stroke="#333" strokeWidth="2"/>

<rect x="185" y="80" width="30" height="25" fill="#FDB5A6" stroke="#333" strokeWidth="1"/>

<rect x="160" y="105" width="80" height="120" rx="5" fill="#FDB5A6" stroke="#333" strokeWidth="2"/>

{/\* Colonna vertebrale \*/}

{layers.skeletal && (

<g>

<line x1="200" y1="90" x2="200" y2="260" stroke="#999" strokeWidth="2" strokeDasharray="3,2"/>

{/\* Vertebre \*/}

{[...Array(12)].map((\_, i) => (

<rect key={i} x="195" y={95 + i \* 14} width="10" height="8" fill="#DDD" stroke="#999" rx="2"/>

))}

</g>

)}

{/\* Scapole \*/}

{layers.skeletal && (

<g opacity="0.5">

<path d="M 165 120 Q 160 140 165 160 L 180 140 Z" fill="#EEE" stroke="#999"/>

<path d="M 235 120 Q 240 140 235 160 L 220 140 Z" fill="#EEE" stroke="#999"/>

</g>

)}

<rect x="110" y="110" width="25" height="100" rx="5" fill="#FDB5A6" stroke="#333" strokeWidth="2" transform="rotate(-5 122 160)"/>

<rect x="265" y="110" width="25" height="100" rx="5" fill="#FDB5A6" stroke="#333" strokeWidth="2" transform="rotate(5 277 160)"/>

<ellipse cx="105" cy="215" rx="15" ry="20" fill="#FDB5A6" stroke="#333" strokeWidth="1"/>

<ellipse cx="295" cy="215" rx="15" ry="20" fill="#FDB5A6" stroke="#333" strokeWidth="1"/>

<rect x="165" y="225" width="70" height="40" rx="5" fill="#FDB5A6" stroke="#333" strokeWidth="2"/>

<rect x="170" y="265" width="28" height="110" rx="5" fill="#FDB5A6" stroke="#333" strokeWidth="2"/>

<rect x="202" y="265" width="28" height="110" rx="5" fill="#FDB5A6" stroke="#333" strokeWidth="2"/>

<ellipse cx="184" cy="385" rx="18" ry="12" fill="#FDB5A6" stroke="#333" strokeWidth="1"/>

<ellipse cx="216" cy="385" rx="18" ry="12" fill="#FDB5A6" stroke="#333" strokeWidth="1"/>

</g>

);

}

return null;

};

return (

<svg width="400" height="450" className="border rounded bg-white">

{getBodyPath()}

{/\* Renderizza marcatori \*/}

{markers.filter(m => m.view === view).map(marker => (

<g key={marker.id} onClick={() => setSelectedMarker(marker)}>

<circle

cx={marker.x}

cy={marker.y}

r="8"

fill={markerTypes[marker.type].color}

fillOpacity="0.6"

stroke={markerTypes[marker.type].color}

strokeWidth="2"

className="cursor-pointer hover:r-10"

/>

<text

x={marker.x}

y={marker.y + 4}

textAnchor="middle"

fill="white"

fontSize="12"

fontWeight="bold"

pointerEvents="none"

>

{markerTypes[marker.type].icon}

</text>

{marker.intensity && (

<text

x={marker.x + 12}

y={marker.y - 8}

fontSize="10"

fill="#333"

>

{marker.intensity}/10

</text>

)}

</g>

))}

{/\* Griglia se attiva \*/}

{layers.grid && (

<g opacity="0.2">

{[...Array(8)].map((\_, i) => (

<line key={`h${i}`} x1="0" y1={i \* 50} x2="400" y2={i \* 50} stroke="#CCC"/>

))}

{[...Array(8)].map((\_, i) => (

<line key={`v${i}`} x1={i \* 50} y1="0" x2={i \* 50} y2="450" stroke="#CCC"/>

))}

</g>

)}

</svg>

);

};

return (

<div className="min-h-screen bg-gray-100 p-4">

<div className="max-w-7xl mx-auto">

{/\* Header \*/}

<div className="bg-white rounded-lg shadow mb-4 p-4">

<div className="flex justify-between items-center">

<h1 className="text-2xl font-bold flex items-center gap-2">

<Activity className="w-8 h-8 text-blue-600" />

Body Mapping - Zone Trattamento

</h1>

<div className="flex items-center gap-4 text-sm">

<span>Paziente: Mario Rossi</span>

<span>Data: {new Date().toLocaleDateString('it-IT')}</span>

<span>Terapista: Dr. Bianchi</span>

</div>

</div>

</div>

<div className="grid grid-cols-4 gap-4">

{/\* Toolbar Sinistra \*/}

<div className="bg-white rounded-lg shadow p-4">

<h3 className="font-semibold mb-3">Strumenti</h3>

{/\* Strumenti di disegno \*/}

<div className="space-y-2 mb-4">

<button

onClick={() => setCurrentTool('marker')}

className={`w-full p-2 rounded flex items-center gap-2 ${

currentTool === 'marker' ? 'bg-blue-100 text-blue-700' : 'hover:bg-gray-100'

}`}

>

<MapPin className="w-4 h-4" />

Marcatore

</button>

<button

onClick={() => setCurrentTool('area')}

className={`w-full p-2 rounded flex items-center gap-2 ${

currentTool === 'area' ? 'bg-blue-100 text-blue-700' : 'hover:bg-gray-100'

}`}

>

<Circle className="w-4 h-4" />

Area Circolare

</button>

<button

onClick={() => setCurrentTool('freehand')}

className={`w-full p-2 rounded flex items-center gap-2 ${

currentTool === 'freehand' ? 'bg-blue-100 text-blue-700' : 'hover:bg-gray-100'

}`}

>

<Pencil className="w-4 h-4" />

Disegno Libero

</button>

<button

onClick={() => setCurrentTool('text')}

className={`w-full p-2 rounded flex items-center gap-2 ${

currentTool === 'text' ? 'bg-blue-100 text-blue-700' : 'hover:bg-gray-100'

}`}

>

<Type className="w-4 h-4" />

Testo

</button>

<button

onClick={() => setCurrentTool('eraser')}

className={`w-full p-2 rounded flex items-center gap-2 ${

currentTool === 'eraser' ? 'bg-blue-100 text-blue-700' : 'hover:bg-gray-100'

}`}

>

<Eraser className="w-4 h-4" />

Gomma

</button>

</div>

{/\* Tipi di marcatore \*/}

{currentTool === 'marker' && (

<div className="border-t pt-3">

<h4 className="text-sm font-medium mb-2">Tipo Marcatore</h4>

<div className="space-y-1">

{Object.entries(markerTypes).map(([key, marker]) => (

<button

key={key}

onClick={() => setMarkerType(key)}

className={`w-full p-2 rounded flex items-center gap-2 text-sm ${

markerType === key ? 'bg-blue-100' : 'hover:bg-gray-50'

}`}

>

<span style={{ color: marker.color }}>{marker.icon}</span>

<span>{marker.name}</span>

</button>

))}

</div>

</div>

)}

{/\* Azioni \*/}

<div className="border-t pt-3 mt-3 space-y-2">

<button

onClick={handleUndo}

className="w-full p-2 rounded flex items-center gap-2 hover:bg-gray-100"

>

<RotateCcw className="w-4 h-4" />

Annulla

</button>

<button

onClick={() => setMarkers([])}

className="w-full p-2 rounded flex items-center gap-2 hover:bg-gray-100 text-red-600"

>

<Trash2 className="w-4 h-4" />

Pulisci Tutto

</button>

</div>

</div>

{/\* Area Canvas Centrale \*/}

<div className="col-span-2 bg-white rounded-lg shadow p-4">

{/\* View Selector \*/}

<div className="flex justify-center gap-2 mb-4">

<button

onClick={() => setCurrentView('anterior')}

className={`px-4 py-2 rounded ${

currentView === 'anterior'

? 'bg-blue-600 text-white'

: 'bg-gray-200 hover:bg-gray-300'

}`}

>

Vista Anteriore

</button>

<button

onClick={() => setCurrentView('posterior')}

className={`px-4 py-2 rounded ${

currentView === 'posterior'

? 'bg-blue-600 text-white'

: 'bg-gray-200 hover:bg-gray-300'

}`}

>

Vista Posteriore

</button>

<button

onClick={() => setCurrentView('lateral')}

className={`px-4 py-2 rounded ${

currentView === 'lateral'

? 'bg-blue-600 text-white'

: 'bg-gray-200 hover:bg-gray-300'

}`}

disabled

>

Vista Laterale

</button>

</div>

{/\* Canvas Area \*/}

<div className="flex justify-center" onClick={handleCanvasClick}>

<BodySVG view={currentView} />

</div>

{/\* Zoom Controls \*/}

<div className="flex justify-center gap-2 mt-4">

<button

onClick={() => setZoom(Math.max(0.5, zoom - 0.1))}

className="p-2 bg-gray-100 rounded hover:bg-gray-200"

>

<ZoomOut className="w-4 h-4" />

</button>

<span className="px-3 py-2">{Math.round(zoom \* 100)}%</span>

<button

onClick={() => setZoom(Math.min(2, zoom + 0.1))}

className="p-2 bg-gray-100 rounded hover:bg-gray-200"

>

<ZoomIn className="w-4 h-4" />

</button>

<button

onClick={() => setZoom(1)}

className="p-2 bg-gray-100 rounded hover:bg-gray-200"

>

<Maximize2 className="w-4 h-4" />

</button>

</div>

{/\* Layer Controls \*/}

<div className="mt-4 p-3 bg-gray-50 rounded">

<div className="flex items-center justify-between mb-2">

<span className="text-sm font-medium">Layer Anatomici</span>

<button

onClick={() => setShowLayers(!showLayers)}

className="text-blue-600 hover:text-blue-800"

>

<Layers className="w-4 h-4" />

</button>

</div>

{showLayers && (

<div className="space-y-1">

<label className="flex items-center gap-2 text-sm">

<input

type="checkbox"

checked={layers.surface}

onChange={(e) => setLayers({...layers, surface: e.target.checked})}

/>

Superficie

</label>

<label className="flex items-center gap-2 text-sm">

<input

type="checkbox"

checked={layers.muscular}

onChange={(e) => setLayers({...layers, muscular: e.target.checked})}

/>

Sistema Muscolare

</label>

<label className="flex items-center gap-2 text-sm">

<input

type="checkbox"

checked={layers.skeletal}

onChange={(e) => setLayers({...layers, skeletal: e.target.checked})}

/>

Sistema Scheletrico

</label>

<label className="flex items-center gap-2 text-sm">

<input

type="checkbox"

checked={layers.nervous}

onChange={(e) => setLayers({...layers, nervous: e.target.checked})}

/>

Sistema Nervoso

</label>

</div>

)}

</div>

</div>

{/\* Pannello Destro \*/}

<div className="space-y-4">

{/\* Template Rapidi \*/}

<div className="bg-white rounded-lg shadow p-4">

<h3 className="font-semibold mb-3">Template Rapidi</h3>

<div className="space-y-2">

{Object.entries(templates).map(([key, template]) => (

<button

key={key}

onClick={() => applyTemplate(key)}

className="w-full p-2 text-left rounded hover:bg-gray-100 flex items-center justify-between"

>

<span className="text-sm">{template.name}</span>

<Target className="w-4 h-4 text-gray-400" />

</button>

))}

</div>

</div>

{/\* Dettagli Marcatore Selezionato \*/}

{selectedMarker && (

<div className="bg-white rounded-lg shadow p-4">

<h3 className="font-semibold mb-3">Dettagli Marcatore</h3>

<div className="space-y-2">

<div className="flex items-center gap-2">

<span style={{ color: markerTypes[selectedMarker.type].color }}>

{markerTypes[selectedMarker.type].icon}

</span>

<span className="text-sm">{markerTypes[selectedMarker.type].name}</span>

</div>

<div>

<label className="text-sm text-gray-600">Intensità</label>

<input

type="range"

min="0"

max="10"

value={selectedMarker.intensity || 5}

onChange={(e) => {

const updated = markers.map(m =>

m.id === selectedMarker.id

? {...m, intensity: parseInt(e.target.value)}

: m

);

setMarkers(updated);

}}

className="w-full"

/>

<div className="flex justify-between text-xs text-gray-500">

<span>0</span>

<span>{selectedMarker.intensity || 5}</span>

<span>10</span>

</div>

</div>

<div>

<label className="text-sm text-gray-600">Note</label>

<textarea

className="w-full p-2 border rounded text-sm"

rows="3"

placeholder="Aggiungi note..."

value={selectedMarker.notes || ''}

onChange={(e) => {

const updated = markers.map(m =>

m.id === selectedMarker.id

? {...m, notes: e.target.value}

: m

);

setMarkers(updated);

}}

/>

</div>

<button

onClick={() => {

setMarkers(markers.filter(m => m.id !== selectedMarker.id));

setSelectedMarker(null);

}}

className="w-full p-2 bg-red-100 text-red-600 rounded hover:bg-red-200"

>

<Trash2 className="w-4 h-4 inline mr-2" />

Elimina Marcatore

</button>

</div>

</div>

)}

{/\* Legenda \*/}

<div className="bg-white rounded-lg shadow p-4">

<h3 className="font-semibold mb-3">Legenda</h3>

<div className="space-y-2">

{Object.entries(markerTypes).slice(0, 4).map(([key, marker]) => (

<div key={key} className="flex items-center gap-2 text-sm">

<span style={{ color: marker.color }}>{marker.icon}</span>

<span>{marker.name}</span>

</div>

))}

</div>

</div>

{/\* Azioni \*/}

<div className="bg-white rounded-lg shadow p-4">

<h3 className="font-semibold mb-3">Azioni</h3>

<div className="space-y-2">

<button className="w-full p-2 bg-blue-600 text-white rounded hover:bg-blue-700 flex items-center justify-center gap-2">

<Save className="w-4 h-4" />

Salva Mappa

</button>

<button className="w-full p-2 border border-gray-300 rounded hover:bg-gray-50 flex items-center justify-center gap-2">

<Printer className="w-4 h-4" />

Stampa

</button>

<button className="w-full p-2 border border-gray-300 rounded hover:bg-gray-50 flex items-center justify-center gap-2">

<Download className="w-4 h-4" />

Esporta PDF

</button>

<button className="w-full p-2 border border-gray-300 rounded hover:bg-gray-50 flex items-center justify-center gap-2">

<Copy className="w-4 h-4" />

Duplica

</button>

</div>

</div>

</div>

</div>

{/\* Riepilogo Marcatori \*/}

{markers.length > 0 && (

<div className="bg-white rounded-lg shadow mt-4 p-4">

<h3 className="font-semibold mb-3">Riepilogo Marcatori ({markers.length})</h3>

<div className="grid grid-cols-2 gap-4">

<div>

<h4 className="text-sm font-medium mb-2">Vista Anteriore</h4>

<div className="space-y-1">

{markers.filter(m => m.view === 'anterior').map(marker => (

<div key={marker.id} className="flex items-center gap-2 text-sm p-1 hover:bg-gray-50 rounded">

<span style={{ color: markerTypes[marker.type].color }}>

{markerTypes[marker.type].icon}

</span>

<span>{markerTypes[marker.type].name}</span>

{marker.intensity && (

<span className="text-gray-500">({marker.intensity}/10)</span>

)}

{marker.notes && (

<span className="text-gray-400 truncate">- {marker.notes}</span>

)}

</div>

))}

{markers.filter(m => m.view === 'anterior').length === 0 && (

<p className="text-gray-400 text-sm">Nessun marcatore</p>

)}

</div>

</div>

<div>

<h4 className="text-sm font-medium mb-2">Vista Posteriore</h4>

<div className="space-y-1">

{markers.filter(m => m.view === 'posterior').map(marker => (

<div key={marker.id} className="flex items-center gap-2 text-sm p-1 hover:bg-gray-50 rounded">

<span style={{ color: markerTypes[marker.type].color }}>

{markerTypes[marker.type].icon}

</span>

<span>{markerTypes[marker.type].name}</span>

{marker.intensity && (

<span className="text-gray-500">({marker.intensity}/10)</span>

)}

{marker.notes && (

<span className="text-gray-400 truncate">- {marker.notes}</span>

)}

</div>

))}

{markers.filter(m => m.view === 'posterior').length === 0 && (

<p className="text-gray-400 text-sm">Nessun marcatore</p>

)}

</div>

</div>

</div>

</div>

)}

{/\* Note Cliniche \*/}

<div className="bg-white rounded-lg shadow mt-4 p-4">

<h3 className="font-semibold mb-3">Note Cliniche</h3>

<textarea

className="w-full p-3 border rounded"

rows="4"

placeholder="Inserisci note cliniche relative alle zone di trattamento..."

defaultValue="Paziente riferisce dolore maggiore al mattino, migliora con il movimento.

Presenza di trigger points attivi nel quadrato dei lombi bilateralmente.

Limitazione funzionale in flessione anteriore (tocca a 20cm dal pavimento).

Piano: focus su release miofasciale e mobilizzazione L4-L5."

/>

</div>

{/\* Footer con info \*/}

<div className="bg-gray-100 rounded-lg mt-4 p-4 text-center text-sm text-gray-600">

<div className="flex justify-center items-center gap-4">

<span className="flex items-center gap-1">

<Info className="w-4 h-4" />

Clicca sulla figura per aggiungere marcatori

</span>

<span>•</span>

<span>Seleziona un marcatore per modificarlo</span>

<span>•</span>

<span>Usa i template per patologie comuni</span>

</div>

</div>

</div>

</div>

);

};

export default BodyMappingInterface;